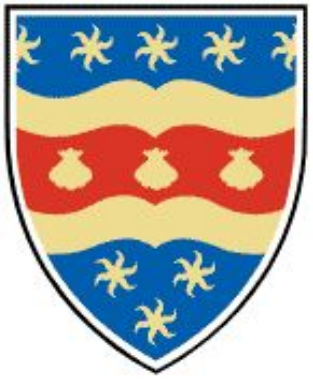
ROCO104

Robot Design and Build

Group A

UNIVERSITY OF PLYMOUTH



Group Members: Rachel Ireland-Jones (10611816), Michael Snelling (10446296), Jake Smart (10562813), Jack Harris (10456011)
Module Leader: Shakil Awan
Submitted: 22 May 2020

Summary

This report provides an insight into the development and design of our bead-pushing-buggy. The brief was to build a buggy that pushed as many beads as possible over the halfway point of an arena in a set time. As we had completed the design and the code was fully written before COVID-19 we were able to fully discuss and test each aspect of the design and test most of it practically. We are confident that our design would have completed this task well if we could have continued our developments. The code was tested with a makeshift buggy and performed as expected, following the pattern we had theorised would be the most efficient for moving as many beads as possible.

KEY STAGES	**PROGRESS**
STAGE 1: Individual Creation of Plymouth Shell	Complete
STAGE 2: Initial Plan for Competition	Complete
STAGE 3: First buggy design	Complete
STAGE 4: Ultrasonic Testing and Coding	Complete
STAGE 5: Real-life test	Complete
STAGE 6: Improvements	Partial
STAGE 7: Final Design Complete with Code	Unachievable
STAGE 8: Competition	Unachievable

In this report, we use engineering drawings, examples of code and external research to put forward our designs and discuss the benefits and drawbacks of our choices.

The report also touches on how COVID-19 affected our progress and limited our design and its implementation. Despite these issues, we did create a functional buggy and so for future developments, we are able to focus on how we could have improved from here if we had more time for the practical work.

Glossary

PLA	Polylactic acid is a fully biodegradable thermoplastic polymer consisting of renewable raw materials. Among all 3D printing materials, PLA is part of the most popular materials used for additive manufacturing.
TPU	Thermoplastic polyurethane is any of a class of polyurethane plastics with many properties, including elasticity, transparency, and resistance to oil, grease and abrasion.
Scoop	The scoop is the front of the buggy that pushes the beads. 3D printed from PLA.
Dolly Wheel	The wheel that the front of the buggy is balanced by.
Skirt	A flexible 3D-printed flap (printed from TPU) that connects to the scoop which can flop over the halfway divider.

Table of Contents

Introduction

The brief we were given was to design a buggy which could push beads over a small ramp in the centre of an arena shown in Figure 1. The measurements of which are approximately 100 x 70 cm, with a wall height of 10 cm). The dividing ramp is approximately 1.5 cm high and 5 cm wide. The buggy is placed in its starting location and given 90 seconds relying solely on the physical design and the code to push as many beads as possible over the ramp.

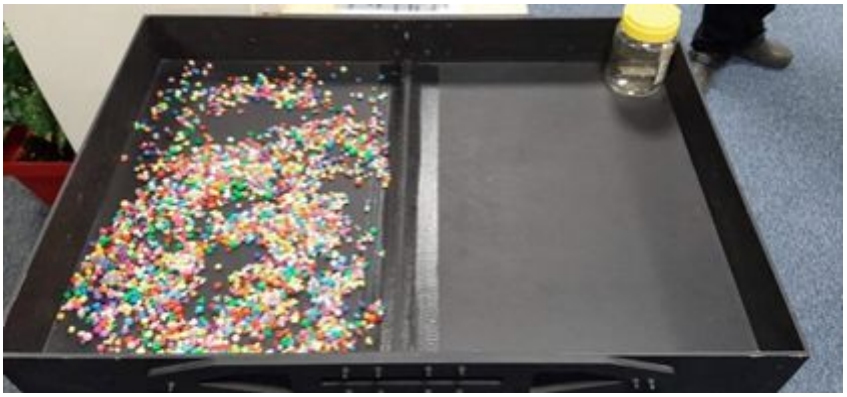


Figure 1 - Beads and the arena

We must keep our design within certain restraints, these being that our robot must fit within an 18 x 18 x 18 cm bounding box and the weight of our design must not exceed 1kg. We can also reset the buggy back to its starting place a maximum of 3 times. Our design incorporates the use of an ultrasonic sensor, two microswitches, two motors and the Plymouth shell that we each built individually.

This report will discuss our design process and our final implementation of ideas as well as how we overcame challenges along the way.

Plymouth Shell Integration and Testing

We tested our Plymouth shell builds by connecting them to our Nucleo boards and loading some test code onto them. The connection is shown below in FIgure 2, the 10, 8 and 6 pin connectors we had individually soldered are used to connect to the long pin connectors of the Nucelo board. The code checked through each component of the Plymouth shell to see if it worked, all our Plymouth shells were successful. In this code we had it set so if one of the switches was closed then it would stop the motors from turning, this checked both the wiring for the switches and motors were correct. We took turns to make sure the ultrasonic sensor was getting a reading when connected to our circuits, also the buzzer was set to play a tune when the blue button was pushed.

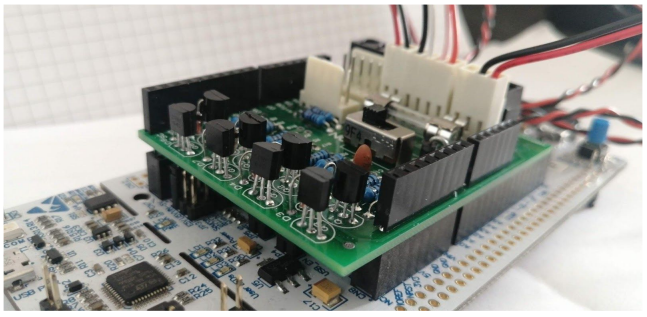


Figure 2 - Plymouth H-Bridge and Nucleo Board together

```
#include "mbed.h"
#include "motor.h"
#include "tunes.h"
#define TIME_PERIOD 2                   //Constant compiler Values here 2 equates to 2ms or 500Hz base Frequency
#define DUTY 0.9                        //DUTY of 1.0=100%, 0.4=40% etc.,

DigitalIn microswitch1(D4);             //Instance of the DigitalIn class called 'microswitch1'
DigitalIn microswitch2(D3);             //Instance of the DigitalIn class called 'microswitch2'
Motor Wheel(D13,D11,D9,D10);            //Instance of the Motor Class called 'Wheel' see motor.h and motor.cpp
DigitalIn myButton(USER_BUTTON);        //USER_BUTTON is the Blue Button on the NUCLEO Board
DigitalOut led(LED3);                   //LED1 is the Green LED on the NUCLEO board
                                        //N.B. The RED LED is the POWER Indicator
                                        //and the Multicoloured LED indicates status of the ST-LINK Programming cycle
Serial pc(USBTX,USBRX);
float duty=DUTY;
int main ( )
{
        pc.baud(9600);                  //BAUD Rate to 9600
        pc.printf("ROCO104 Demonstration Robot Buggy Plymouth University 2018/19\n\r");

        Wheel.Period_in_ms(TIME_PERIOD);            //Set frequency of the PWMs

        Wheel.Stop();

        close_encounter(1);             //tune to play Announce start!

        while(myButton==0)
        {       //Wait here for USER Button (Blue) on Nucleo Board (goes to zero when pressed)
                led=0;                  //and flash green LED whilst waiting
                wait(0.1);
                led=1;
                wait(0.1);
        }
```

```
        Wheel.Period_in_ms(2);              //Set frequency of the PWMs 500Hz
        while(true)                         //Repeat the following forever NB always true!
        {
                Wheel.Speed(0.8,0.8);       //Forward 80%
                RevStop();
        }
}
```

The written code above was used to test every individual aspect of our circuits to see if they were performing correctly, “Wheel.Stop()” sets the motors to be off until the blue button is pushed, and “Wheel.Period_in_ms(2)” sets the frequency for the pulse width modulation to 500Hz. Once the button is pushed a while loop begins with “Wheel.Speed(0.8,0.8)” which sets the power of the motors to 80%, this would continue until either of the microswitches were closed calling the function “RevStop()”, stopping the motors until released.

Later on in the process, we had written code which incorporated the use of the ultrasonic sensor, it was at this point which we individually connected the sensor to our boards to ensure the wiring was correct. Everyone managed to have the ultrasonic sensor work without fault and using PuTTY to show us the readings we checked that the sensor was accurately calculating the distance.

Design and Simulations

Plan for Competition

The first thing we discussed was the plan for how we wanted our buggy to move around the arena and the best way to move the most beads in the shortest amount of time. We knew that whatever we decided for the movement would hugely impact the design of the buggy itself and so it had to be the first thing we did.

Due to the fact the brief specified we could start with the buggy in any position in the arena, we chose to place the buggy in the leftmost back corner as this meant that hypothetically we only had to move in one direction and so would need fewer sensors or microswitch.

At this point the brief hadn't been finalised meaning we weren't aware of the fact we could re-place the buggy up to three times during the competition and so devised a plan we hoped would clear as many beads as possible in one go.

The plan for the buggy's movement pattern involved using the second microswitch on the right side of the chassis. When the switch was pressed the code would call another function in which the only difference in the movement instructions was the direction of the turns, sending it back toward it's starting position sweeping the arena on its way. This would effectively reduce the number of times we would need to reset the position of the buggy as it would sweep the arena 10 times before coming to a stop.

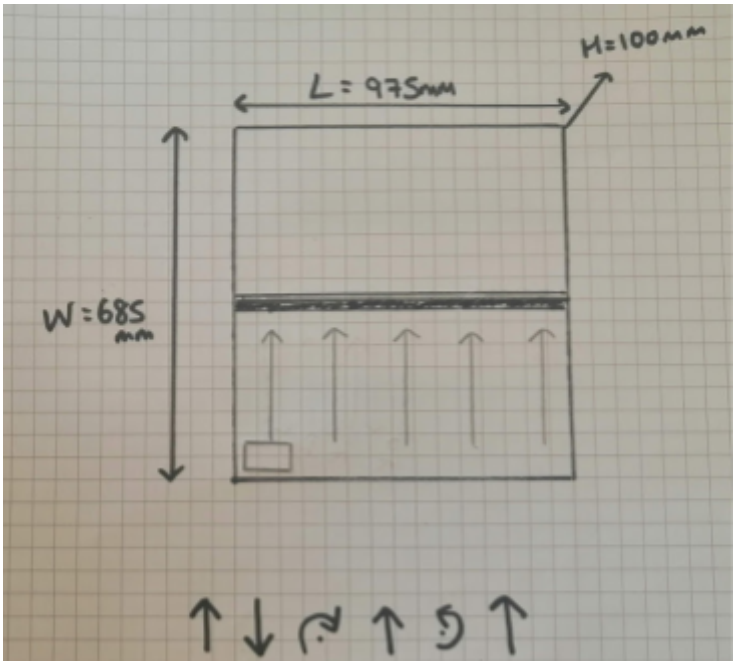


Figure 3 - Plan of action.

We measured the arena and using the size constraints from the brief, we decided to have the buggy perform the same movements 5 times at different points across the board as drawn in

Figure 3. As our buggy can only move forwards and backwards as it has no axle, we also added symbols to remind us when starting the coding how we planned to move.

Initial Design

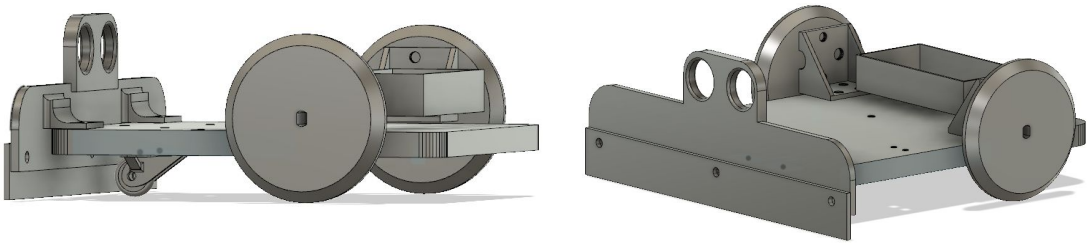


Figure 4 - Initial Design

With the competition plan in mind, we started planning out the buggy design. We knew the beads in the arena were around 2-4mm cubed meaning that whichever method we chose to push the beads would need to be able to push them around and not just roll over them. Our initial design is shown in Figure 4, it doesn't have the elements provided by the university in it however all of the aspects we initially planned to design and make ourselves are included.

To do this we decided to create a two-part pusher, one solid for strength and one flexible at the bottom so that our buggy did not get caught on the ramp and fail in the competition. We called these the scoop and the skirt respectively from here on. This combination effect is shown in Figure 5, the scoop is the higher section and the skirt the lower.

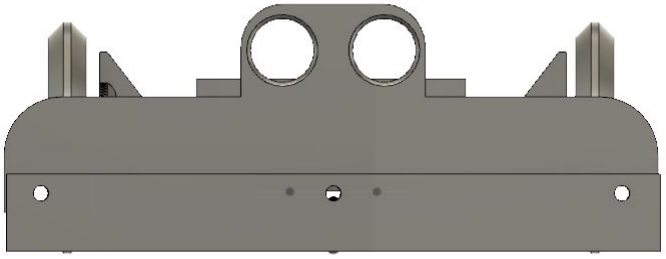


Figure 5 - Initial Scoop and Ultrasonic Holder Design

The scoop we decided would be made of PLA and the skirt would be made of 1mm thick TPU. These would both be 3-D printed and we designed them to overlap by 10mm so they could be joined in three places along the length with screws. With our design, the ultrasonic sensor is located at the front of the buggy with the transceivers sticking out from the holes in the scoop.

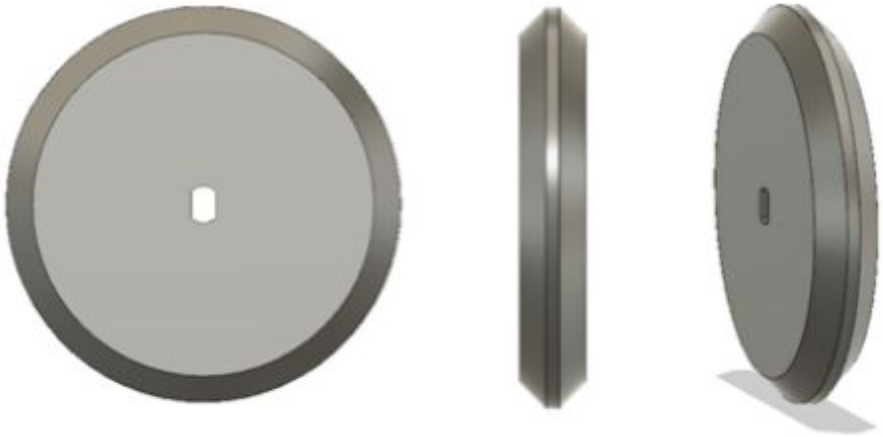


Figure 6 - Initial Wheel Design

Our initial wheel design, as shown in figure 6, came from the thought that we wanted to avoid driving the buggy over any beads and getting stuck or sliding off course. We chose this tapered style to give the least contact with the floor and reduce this possibility however we realised quickly that we would need to continue adapting the design so we had more grip as our initial prototype slipped badly in the arena.

To hold the battery pack, we decided to create a box which we could put on the back of the chassis by the wheels. The initial design on fusion used very thin walls and was to be 3-D printed however our prototype version warped badly and so we realised we would need to make it out of a thicker material. We also realised that the current position we had it in would hinder the use of the board and so we would need to change the positioning. As can be seen in figure 6, we did not originally design a way for it to be connected to the chassis and so we had to develop this in later designs.

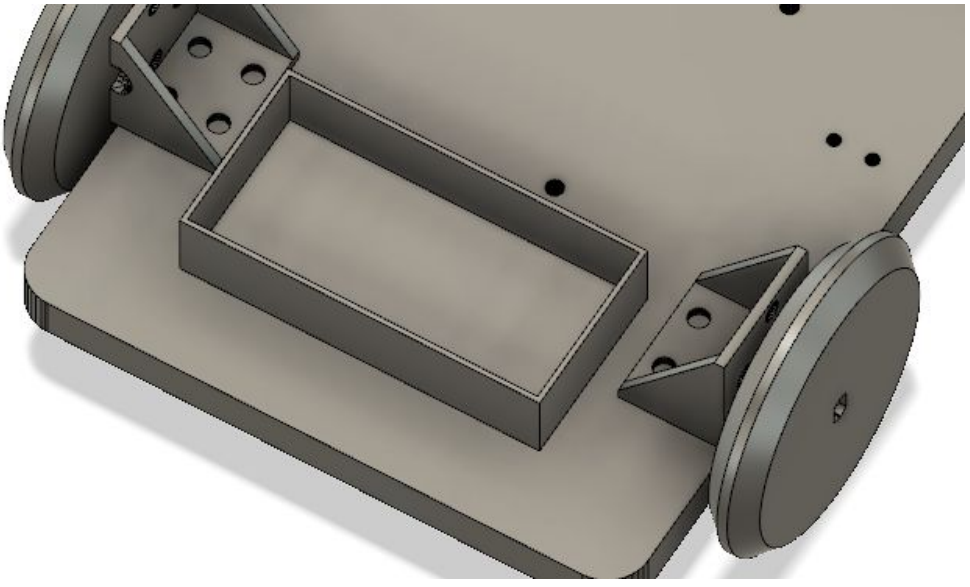


Figure 7 - Initial Battery Holder Design

We also created this schematic to give us an understanding of our hardware. It is the full drawing of the entire Plymouth Shell H-Bridge and shown below in Figure 7. This took some time to create but meant if we had an issue with hardware we could look back and remind ourselves of the individual components and try and understand our issue and form a solution.

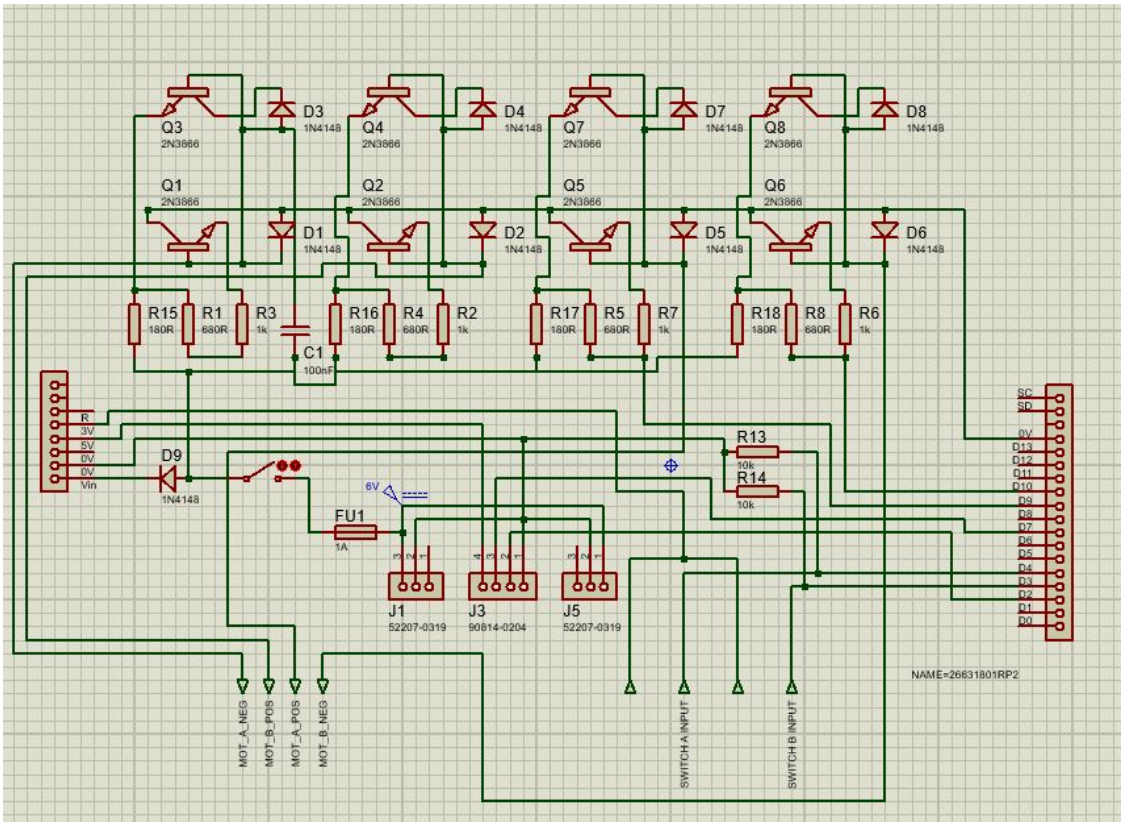


Figure 20 - Schematic Diagram of Plymouth H-Bridge Shell

Final Design

Throughout the design process, we made 32 different versions on Fusion 360 and many different iterations on Solidworks and so understandably our final design had changed a lot from the initial design. We made it more aesthetically pleasing whilst never reducing function and updated individual elements to increase the practicality. Four views of this final design are shown in Figure 8 showing all the individual elements in situ.

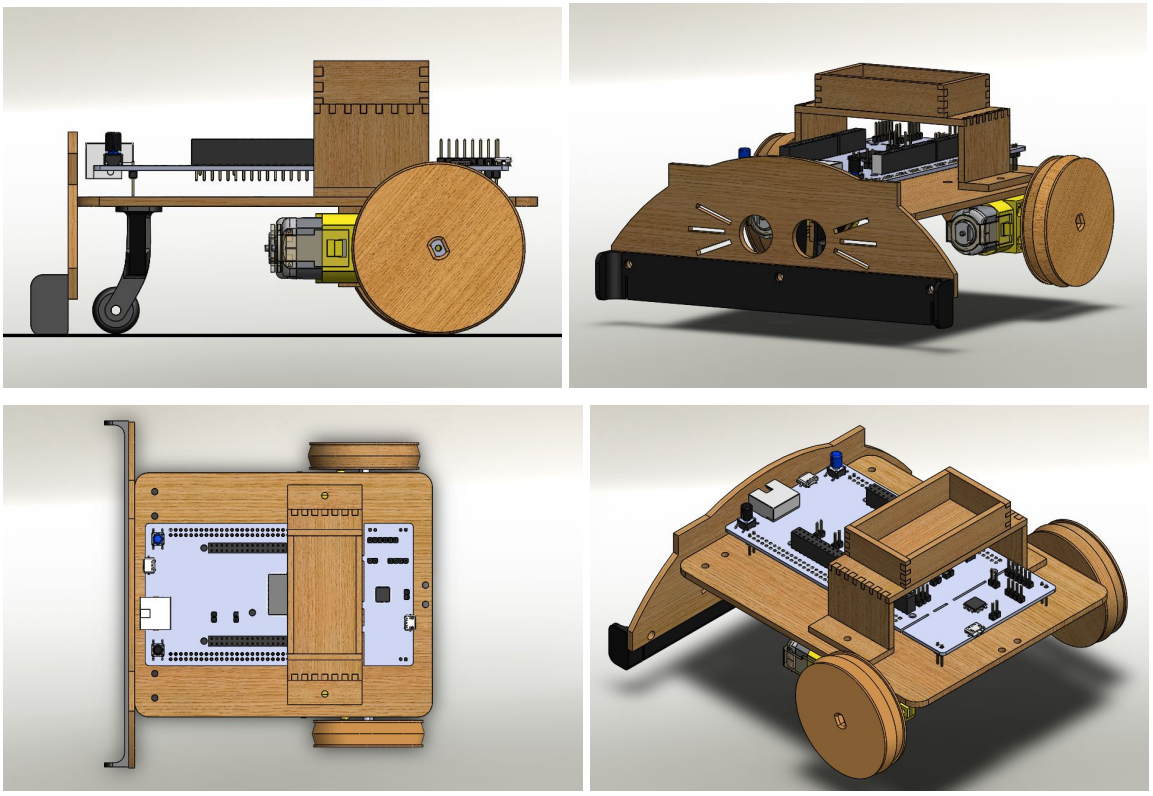


Figure 8 - Final Design and Render

With this final design, the ultrasonic sensors are still on the front however they are lowered slightly so that we could ensure that the echo reflected off the back wall. We decided to make the design more aesthetic by using the holes for the ultrasound as eyes for a cat. (Figure 9) The trigger sends out ultrasonic sound waves which when reflected off a surface back to the echo will calculate the distance from the sensor to the surface by the time taken for the sound wave to travel to the far wall and back.

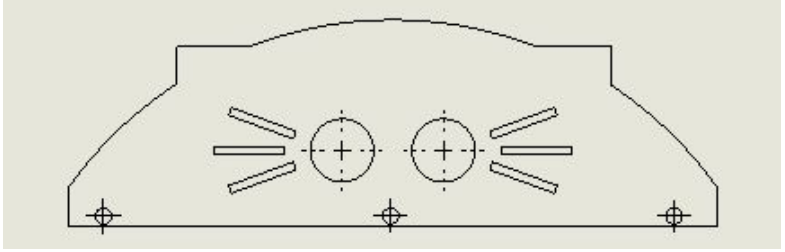


Figure 9 - Final Scoop Design

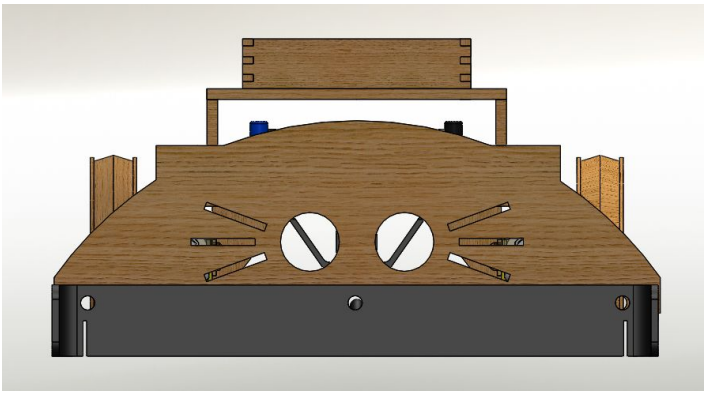


Figure 10 - Front View of the Buggy showing the final scoop and skirt design.

The distance that has been calculated by the code will then move the buggy forward or backwards according to this value of distance. If the distance to the edge of the arena is less than 500mm then the buggy will be made to reverse until the microswitch on the back of the buggy hits the wall. Alternatively, if the distance is more than 500mm when travelling forward (towards the ramp) then the buggy will continue forwards until the 500mm limit is met at which point the reverse process begins.

We designed our battery holder to go over the board so we could make our design as compact and efficient as possible. The batteries would be placed slightly in front of the wheels, the extra weight provided by the batteries would increase the steadiness and grip of the wheels as they travel up and down the arena. The box where the batteries would be housed has been designed to firmly secure the batteries in place to prevent them from sliding around during movement.

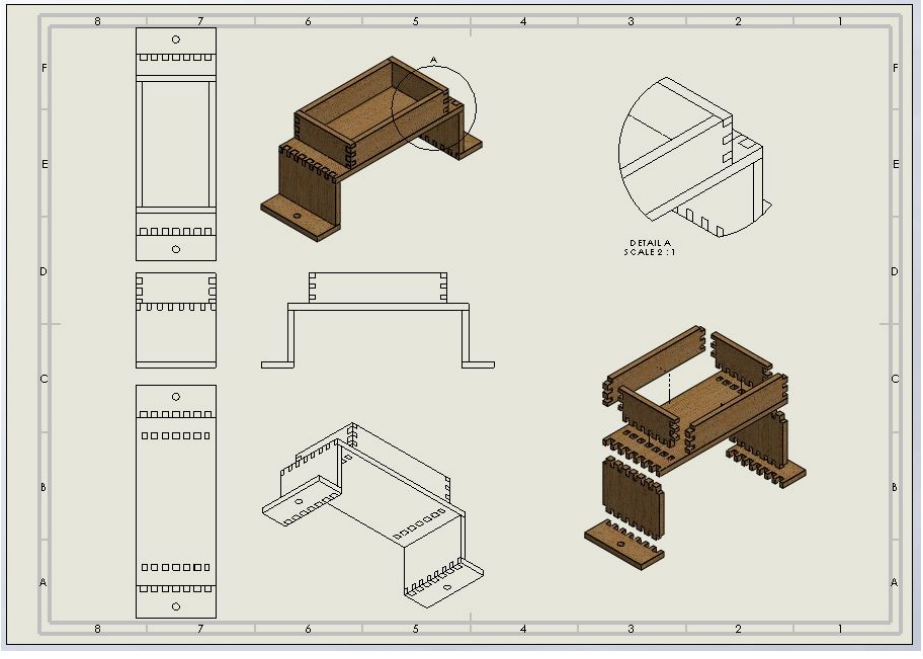


Figure 11 - Battery Holder, engineer drawing and exploded diagram

As we could no longer 3-D print parts, we used tab and slot design so that we could laser cut each individual piece and put them together, still creating a secure hold on the battery and keep the design looking sleek. The full engineer drawing of it, including an exploded drawing showing the tab and slots can be seen above in Figure 11..

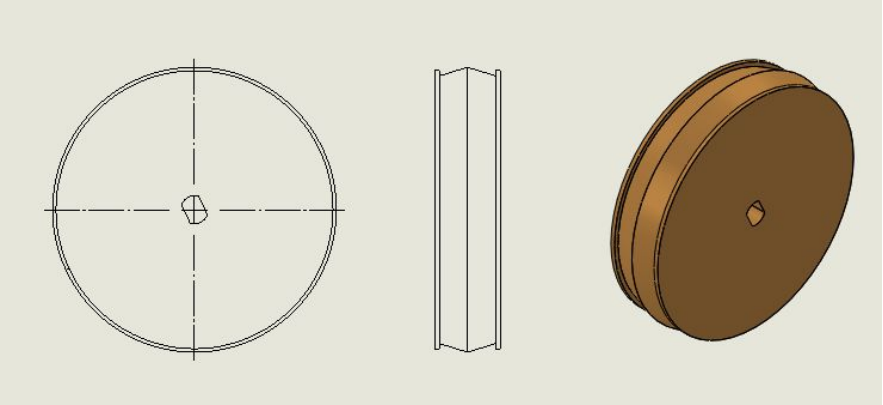


Figure 12 - Final Wheel Design

The wheels are designed with a taper in the middle to push the beads away, the outside of the wheel has a 'lip' on each side to prevent the rubber band from coming off. (Figure 12) The rubber band which fits around the wheel provides grip. We carefully measured and designed the wheels to fit perfectly on the provided motor to create a snug fit which would not wobble or slip, this is shown in Figure 13. In this image you can see the small lip that stops the rubber sliding off.

Figure 13 - Test wheel with rubber for grip.

The DC motors are held on the underside of the buggy using a brace, by having the motors attached from underneath the chassis we free up space on top that we would otherwise have had to use keeping our design compact (figure 14). The motors are being run at 65% power because if it was set to more than this then too much current would be drawn causing the ultrasonic sensor to stop functioning properly, measuring less than 500mm distance prematurely causing the buggy to reverse before reaching the ramp.

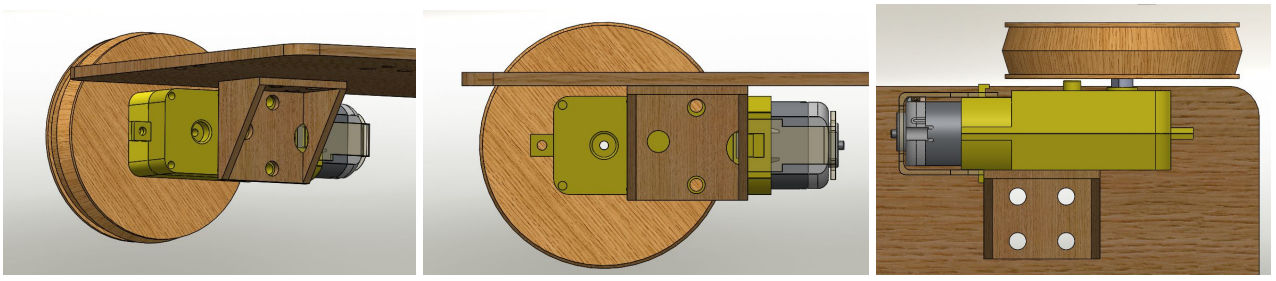


Figure 14 - Motor Brace in Position

Our chassis design here hasn't changed much from our original except for the fact that the elements that connect to it have adapted and moved meaning the placement of certain holes have also changed. Due to the new condition of having no 3-D printed parts, we also adapted the design so it was tab and slot with the scoop which could still be laser cut easily as shown in figures 15a and b.

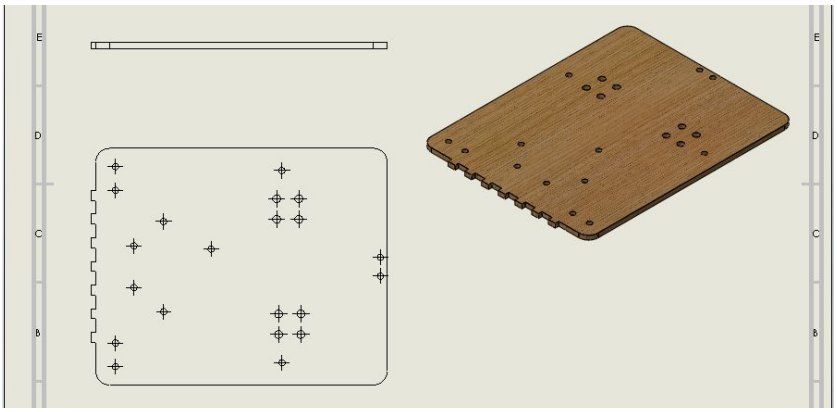


Figure 15a - Final Chassis Design

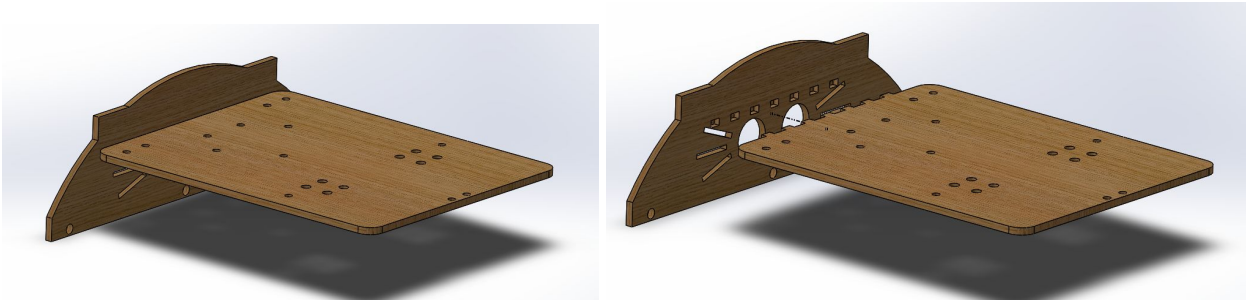


Figure 15b - Chassis and Scoop, Tab and Slot

Build and Test

Our first test of the code was with a makeshift buggy using 3-D printed test parts in the lab and some elastic bands holding everything together. The aim was just to see if the code would work as intended. It was a successful test. The buggy and all components performed exactly as expected meaning the code was complete and correct. However we did not have time to code the switch on the front off side. This switch would've been used to reverse the pattern for which the buggy moved, making it go from right to left instead of the left to right when reaching the right wall of the arena.

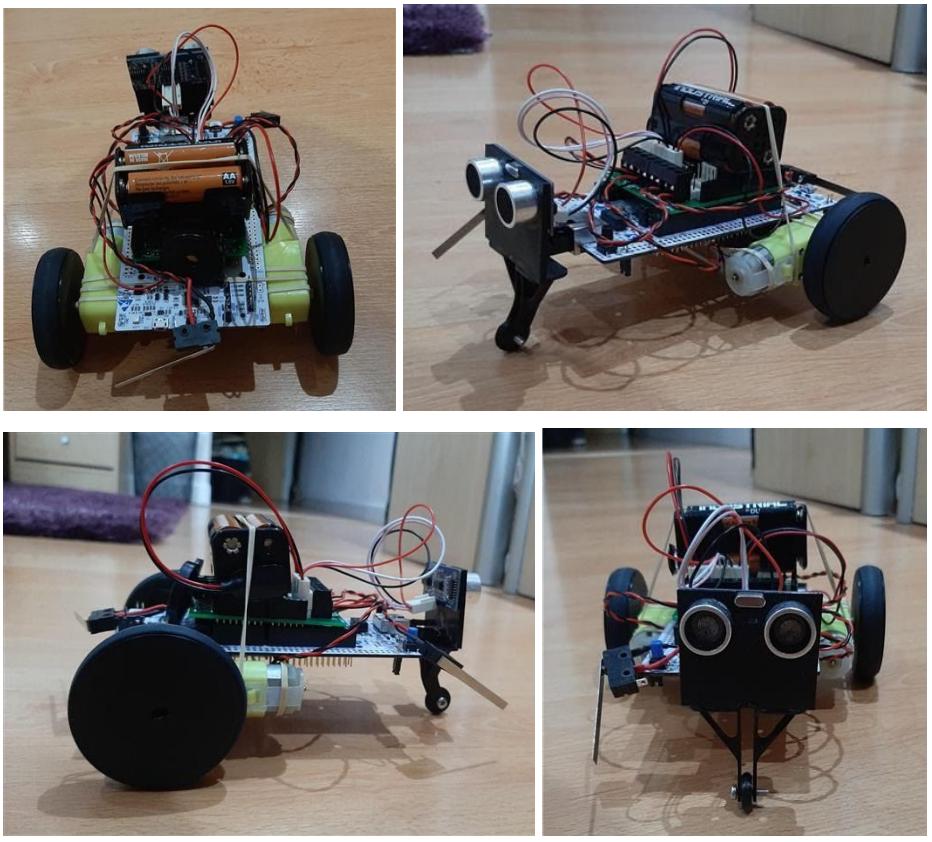


Figure 16- Makeshift Buggy from 3-D printed Parts

For the purpose of testing, we did not attach a scoop to the buggy as we were simply testing the code with regards to movement and not how well the buggy performed at moving beads. Figure 16 shows the makeshift buggy we used for the test. We completed this test the day before labs shut for the foreseeable future and at this time there had been no elements laser cut. Should we have more time in the labs, we would have had our chassis and scoop laser cut so that we could test the buggy in the arena. From here we would have been able to do more testing, possibly making further adjustments to the code and buggy design for maximum efficiency.

Figure 17 - Screenshots from a video of our first test.

Although the quality isn't fantastic we did take a video of the buggy performing its first test. As can be seen in the screenshots the robot does follow the path we have described. (Figure 17).

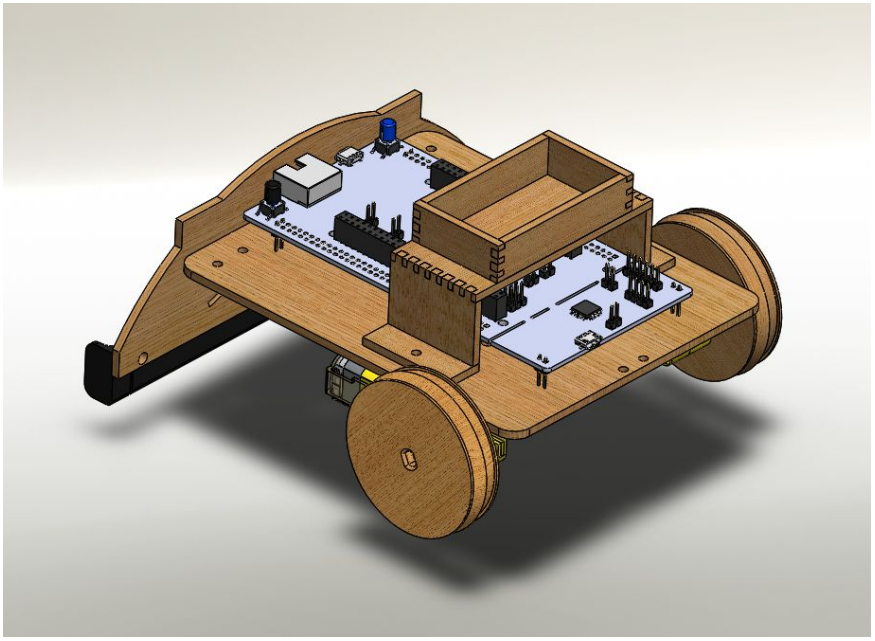


Figure 18 - The final design which couldn't be produced and tested due to COVID-19..

Results

Had we been given the opportunity to complete and demonstrate our design we are confident that it would have been a success but due to COVID-19, this was not possible. As previously mentioned in our build and test section of this report, the code was nearly 100% complete other than one aspect being the front off side microswitch and CAD design being fully complete (figure 19) assuming that no changes had to be made should we be able to test in the area.

With a makeshift buggy constructed using 3D printed parts, we tested the code to make sure it would follow the instructions to go forward until <500mm from the wall, reverse until the rear microswitch hit the wall, then turn and position itself to move forward again. This was performed spectacularly by our test buggy, with no issues that we could see. Observing how the buggy moved with the code implemented, solidified that our approach to how the buggy should maneuver would be the most efficient.

The tests carried out resulted in the confirmation that our code would work with the final design had we been able to get the parts cut and assembled.

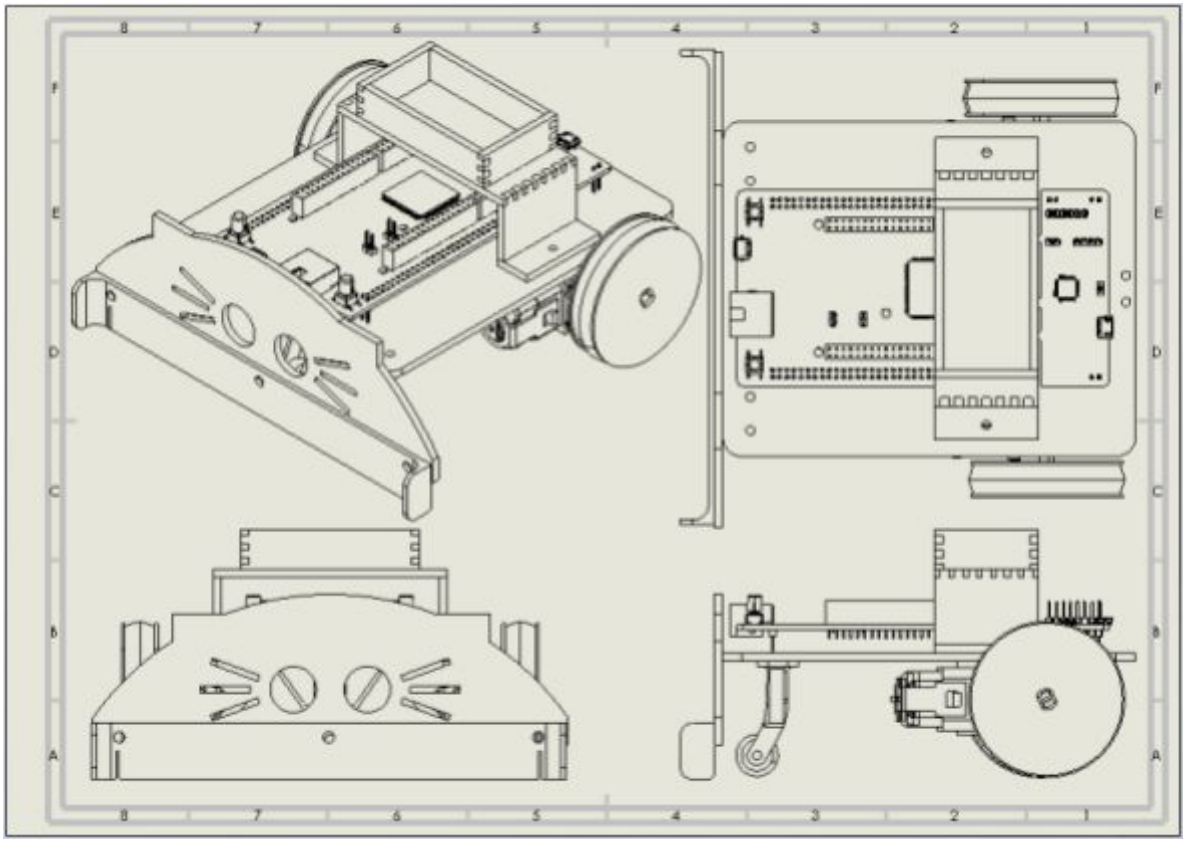


Figure 19 - Final Design and Engineer Drawing

Conclusions & Future Developments

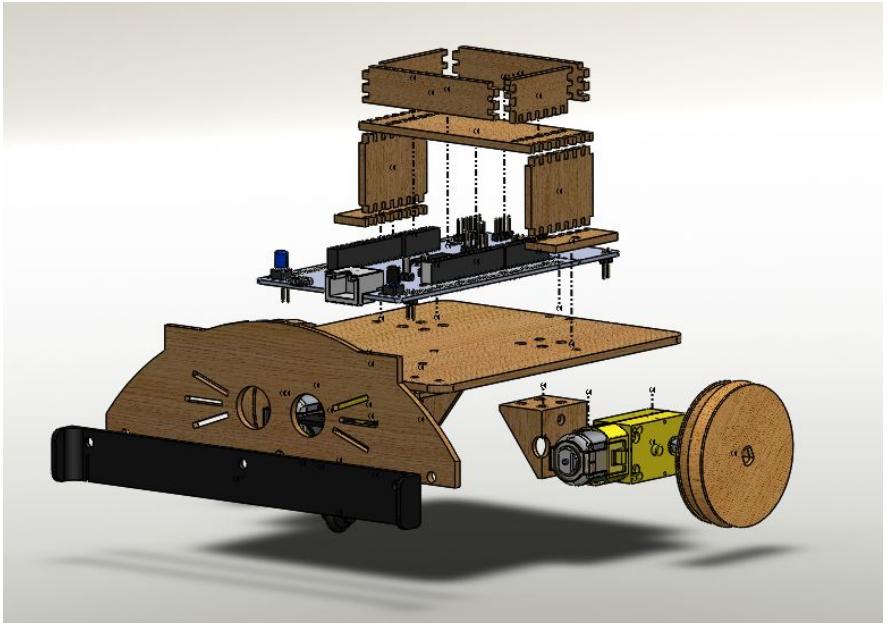


Figure 20 - Final Exploded Design and Render

In conclusion, the design of our buggy and the code would be a success if allowed to attempt the task, the use of the switches and ultrasonic sensor to determine the direction of travel would have been very effective at pushing as many beads as possible over the ramp in the middle of the arena.

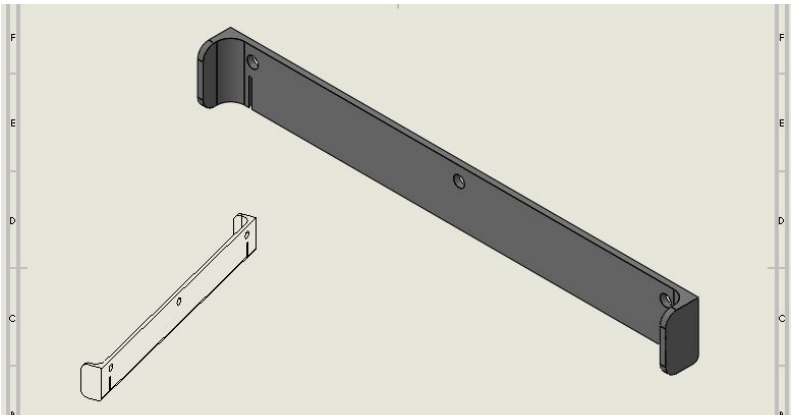


Figure 21 - Skirt Design

If we had been able to continue building and testing our design one change we would possibly have made would be to the design of our scoop as we hadn't had much chance to test the current design shown in Figure 21. With further testing we would be able to determine whether the protruding ends of the skirt would hinder the buggy's ability to push the beads over the ramp by coming into contact with the ramp. If the protruding ends did interfere with this action, then we would most likely change the design to a flat skirt.

Something we would like to have tested was the possibility of integrating the dolly wheel with the scoop to bring the chassis lower to the ground to see if this would have any benefits on the performance.

We would have liked to include the addition of mudguards to prevent beads from going under the wheels of the buggy which could change the direction of travel, knocking it off the course leading to task failure. Mudguards would ensure any of the beads missed by the scoop are pushed to the side of the wheels.

References

[1] Media.digikey.com. 2020. *DC Motor Datasheet*. [online] Available at: <https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf> [Accessed 21 May 2020].

[2] Snelling, M., Ireland-Jones, R. and Smart, J., 2020. *ROCO104_2020_Flowchart_Groupa.Pdf*. [online] PDF. Available at: <https://drive.google.com/open?id=1AJEwBgfkoVYqM16rkG8BuzaZLFvftIwR> [Accessed 21 May 2020].

[3] Snelling, M., Ireland-Jones, R. and Smart, J., 2020. *ROCO104_Groupa - Bead Pushing Buggy*. [online] Os.mbed.com. Available at: <https://os.mbed.com/teams/MRJJ/code/ROCO104_Buggy/> [Accessed 21 May 2020].

Appendices

Data Sheets

Ultrasonic Sensor

TruSens

HC-SR504 Ultrasonic Ranging Module

Order code: **74-1109**

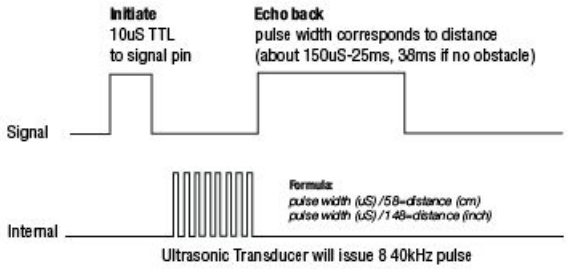
VCC: **5V power supply**
Trig: **Trigger pin**
Echo: **Receive pin**
GND: **Power ground**

Features:

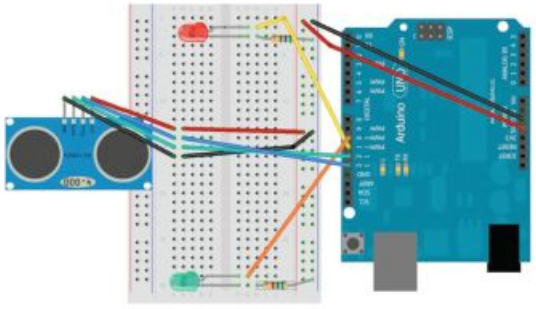
- Power Supply: +5V DC
- Quiescent Current: <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance: 2cm – 400 cm
- Resolution: 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS
- Dimension: 45 x 20 x 15mm

Operation:
The timing diagram of HC-SR04 is shown. To start measurement, Trig of SR04 must receive a pulse of high (5V) for at least 10us, this will initiate the sensor will transmit out 8 cycle of ultrasonic burst at 40kHz and wait for the reflected ultrasonic burst. When the sensor detected ultrasonic from receiver, it will set the Echo pin to high (5V) and delay for a period (width) which proportion to distance. To obtain the distance, measure the width (Ton) of Echo pin.

Time = Width of Echo pulse, in uS (micro second)
Distance in centimeters = Time / 58
Distance in inches = Time / 148
Or you can utilize the speed of sound, which is 340m/s

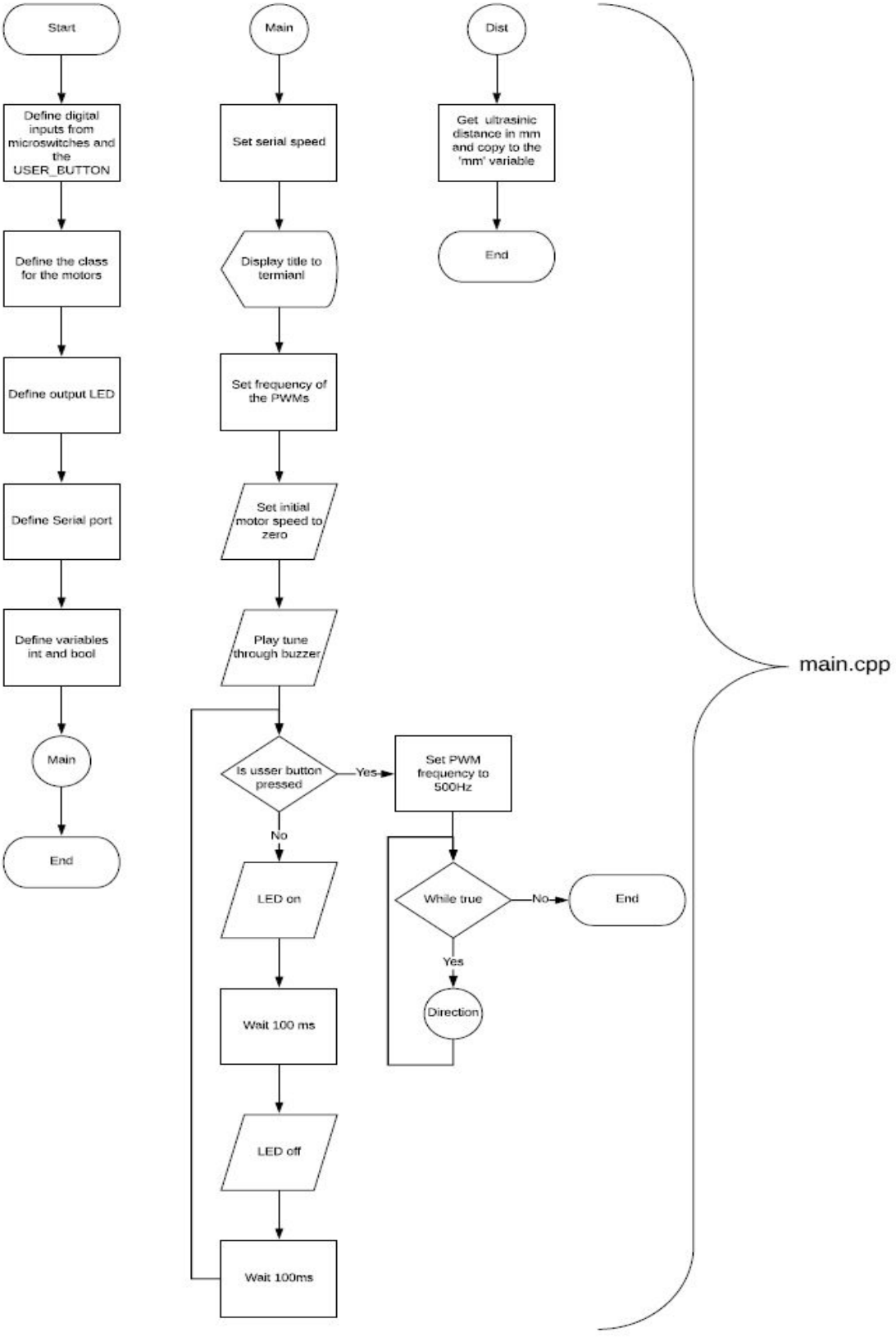


Note:
Please connect the GND pin first before supplying power to VCC.
Please make sure the surface of object to be detect should have at least 0.5 meter2 for better performance.



Flowcharts

Here we have the flowchart for our main.cpp code, the full flowchart can be accessed from the references.



Code

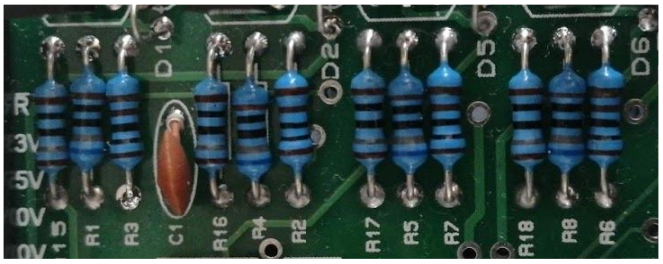
Here we have the code for our main.cpp file, the full code can be accessed from the references.

```
#include "mbed.h"
#include "motor.h"
#include "UltraSonic.h"
#include "tunes.h"
#include "clickers.h"
#include "Movement.h"
#define TIME_PERIOD 2                   //Constant compiler Values here 2 equates to 2ms or 500Hz base Frequency
#define DUTY 0.9                        //DUTY of 1.0=100%, 0.4=40% etc.,

DigitalIn microswitch1(D4);             //Instance of the DigitalIn class called 'microswitch1'
DigitalIn microswitch2(D3);             //Instance of the DigitalIn class called 'microswitch2'
Motor Wheel(D13,D11,D9,D10);            //Instance of the Motor Class called 'Wheel' see motor.h and motor.cpp
DigitalIn myButton(USER_BUTTON);        //USER_BUTTON is the Blue Button on the NUCLEO Board
DigitalOut led(LED3);                   //LED1 is the Green LED on the NUCLEO board
                                        //N.B. The RED LED is the POWER Indicator
                                        //and the Multicoloured LED indicates status of the ST-LINK Programming  cycle

Serial pc(USBTX,USBRX);

/*Instance of the Serial class to enable much faster BAUD rates than standard 9600 i.e 115200. This is Pseudo RS232 over
USB the NUCLEO will appear as a COMx Port, see device Manager on PC. Use PuTTY to monitor check COMx and BAUD rate
(115200)*/

int mm;                         //This variable will hold the distance in mm between the buggy and the object in front of it
bool fStopped, rStopped = false;        // these variables are used to determine whether the buggy has been stopped, either
                                        from going forwards or backwards

void dist () {                          //This function will get the distance in mm and set the mm variable to this value
        ultra_sonic_distance();
        mm = (int)GetDistance();
}
int main () {
        pc.baud(9600);              //BAUD Rate to 9600
        pc.printf("ROCO104 Demonstration Robot Buggy Plymouth University 2018/19\n\r");
        Wheel.Period_in_ms(TIME_PERIOD);//Set frequency of the PWMs
         Wheel.Stop ();

        close_encounter(1);             //tune to play Announce start!

        while(myButton==0)
        {                               //Wait here for USER Button (Blue) on Nucleo Board (goes to zero when pressed)
         led=0;                         //and flash green LED whilst waiting
         wait(0.1);
         led=1;
         wait(0.1);                     //Test Microswitches with two different tones see tunes.cpp tunes.h or flash (led1 and
                                        led2)  onboard LEDs
        }
        Wheel.Period_in_ms(2);          //Set frequency of the PWMs 500Hz

        while(true)                     //Repeat the following forever NB always true!
        {
         Direction();                   // run the Direction function in the Movement library
        }                               // go back to start of while loop
}                                       //end of int main()
```

Rachel Ireland-Jones

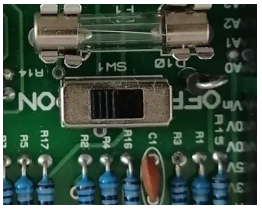
The first thing I did for my Plymouth Shell was to study the instructions, circuit board and components. I then wrote out the list of components and carefully placed the components next to their name, checking each resistor value against the resistor code and with a multimeter as I went.

The first items I placed on my board were the first four resistors. Although the instructions said to do them three at a time, this required me to place three different values at once and having read the instructions I knew that I could place all the resistors of the same value at once. I started with the 680R resistor and placed one in the middle position of each set of four, split the legs and soldered them carefully. I then did the same with the 180R resistors which went to the left and the 1K which went to the right. This gave me my final resistors looking like so.

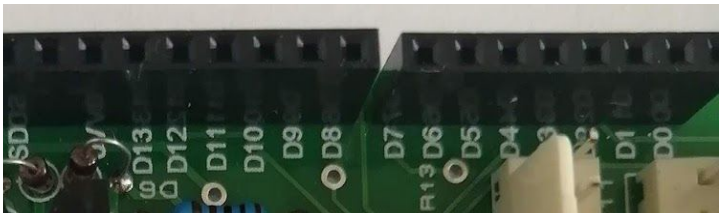


Once I was happy with the results of my soldering and that I had made no mistakes I cut the excess legs off the back to make it neat. Next, I found my ceramic 100nF capacitor, as the legs were already in parallel and I didn't have to worry about polarisation. It was quick and easy to put this in position and solder safely.

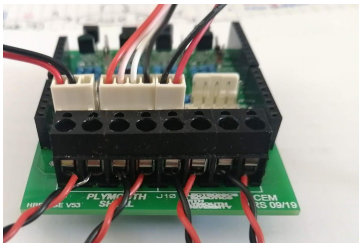
My next task was to position the switch. Although the instructions didn't state if there was any need for the switch to face a specific way, and I didn't think it would be polarised I still made sure to place it the same way around that it was shown in the instructions.



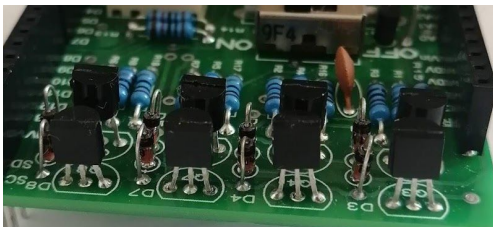
Once I had the switch in place I tackled one of the more difficult sections. The 6,8 and 10-pin connectors. Due to their tall and thin nature, these are very easy to solder at a wonky angle which brings problems when then connecting the pins to another device. I was careful and placed the pins, soldering in just one place and then adjusting as necessary to make them straight before continuing and soldering the rest of the pins. These did not need to be cut as they would connect the Plymouth Shell to the Nucleo Board later.

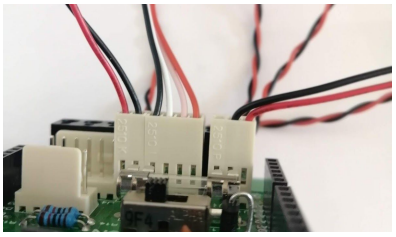


Next, I moved onto the 4-way PCB terminals. These were some of the easiest components on the board as they were the opposite to the pin connectors, short and wide!

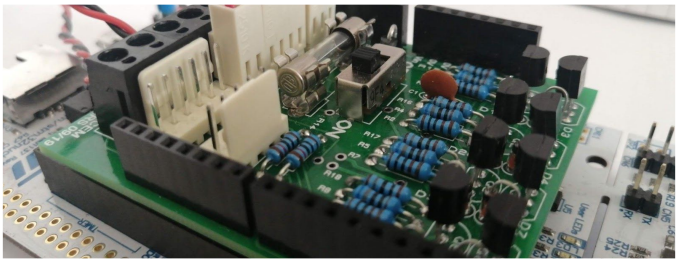


After this, it was time for the NPN BC639 and PNP BC640 transistors. There were four of each of these and it was essential I did not mix them up. Thanks to my organisation earlier this was simple. They held themselves up easily and so were easy to solder once in place. The diodes next were also slightly fiddly as they required bending and positioning. I did this using pliers to ensure they were uniform and correct. I then had to ensure I had them facing the correct way as they are a polarised component. I took my time and was careful so once they were in the correct position it was easy to fix them down.



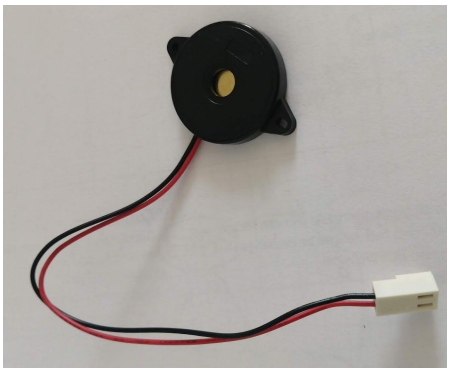


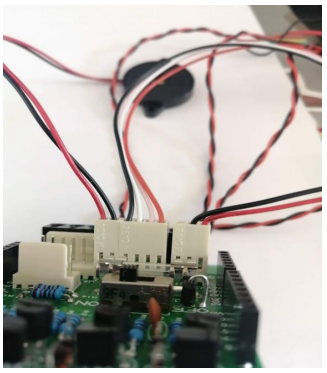
Next was the white MOLEX connectors, these were a little fiddly as they are quite short and when the board was turned over to solder they fell through again. However, by attaching their female counterparts while I soldered I made them taller and fixed this issue. I then soldered the final diode, again being careful of the polarisation as before, this was a simple component. I lastly soldered the fuse holders onto the board again using the pliers so I didn't feel the heat from the metal as I held them in place. This was my final board with all elements soldered.



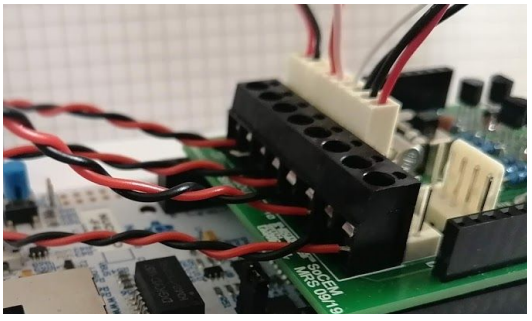
Next, I used some wire cutters to cut lengths of different coloured 20cm wire and then stripped about 8mm off the end of each. Then using a crimping machine I crimped the ends of

them, where necessary to secure the connector to the wire. I had to do this for the buzzer, the battery pack and the ultrasonic connectors. Then I had to push the crimped wire into the female MOLEX counterpart.





To solder, the motors and microswitches used a bit of dexterity and precision as I had to be careful not to melt the plastic nearby and also create a solid connection with stranded wire. For the microswitches, I also put a piece of heat shrink tubing over the wire before soldering and then used a heat gun to shrink the tubing and protect the wire when finished.



The last thing I did was screw the opposite ends of the wires for the microswitches and motors into the PCB screw terminals to connect them up. I then placed the H-Bridge onto the Nucleo F429ZI board to finish.